University Of Bahrain
IT College, Computer Science Dept.
ITCS 215 Data Structures in C++
Quiz#1 Chapter 2: Inheritance & Composition

ID

Name: _

Section:

```
class Printer
{
private:
       string        Origin;
       float         Version;
public:
       Printer( ) {
              Origin="Hewlet Packard";
              Version=1.0;
              cout<<Origin;
       }

       Printer(string   Org, float Ver ) {
              Origin  = Org;
              Version = Ver;
       }

       void print ( ) {
              cout<<"LP: "<< Origin <<
              cout<<Version << endl;
       }
};


class  Laser: public Printer
{
protected:
       bool          color;
       int           pages;
public:

       Laser(string ,float , bool , int );
       void print();

};
```

Implement the following functions:

1) The Laser *constructor* function with 4 parameters.

```
laser::laser(string o, float v, bool c, int p):
Printer(o,v)
{ color = c;
  pages = p;
}
```

2) The print function in class Laser that prints all attributes

```
void Laser::print( )
{ printer::print( );
  cout<<color<<pages;
}
```

3) The output of the following

```
void  main ( )
{
   Printer        HP;


}
```

Hewlet Packard

University Of Bahrain *IT College, Computer Science Dept.* *ITCS 215 Data Structure* *Quiz#2 (Chapter 3)*	ID : _ _ _ _ _ Name: _ Section:

9½

Write a function (not a member function) called **funcL** that accepts an arrayListType object called ***L*** as parameter. The function finds the sum of all even elements in list ***L***, removes them, and finally adds the sum at the end of list ***L***. If the list ***L*** is empty display a msg and end the function.

(You may use functions listsize, retrieveAt , removeAt, insertAt, replaceAt, operator=).

```
template<class Type>
void    funcL ( arrayListType<Type>&  L )
{  Type item, sum;
   sum=0;
   if( L.isEmpty())
   cout << "The function is Empty";
   else
   {  for(int i=0 ; i< L.listsize(); i++)
      {  L.retrieveAt(i, item);
         if( item%2==0 )
         {  sum= sum + item;
            L.removeAt(i);
            i--;
         }
      }

      L.insertEnd(sum);
   }
}
```

University Of Bahrain *IT College, Computer Science Dept.* *ITCS 215 Data Structures* *Quiz#3 Chapter 5*	ID: Name: _ Section: 10

Write a function ***distribute*** to be included as a member function in class ***linkedListType***, which accepts an array and its size as parameters. The function will insert all **odd** numbers in the array at the beginning of the linked list and all **even** numbers at the end of the linked list. (assume the linkedList is not empty)

Function Prototype:

```
void distribute(const Type array[], int size);
```

Do not call any member function of class *linkedListType* in your member function.

```
void linkedListType<Type>::distribute(const Type array[], int size)
{
  nodeType<Type> *newnode;
  newnode = new nodeType<Type>;
  for(int i=0; i<size; i++)
  {
    if (array[i]%2 == 0)
    {
      newnode->info = array[i];
      newnode->linke = NULL;
      if (first==NULL)
      {
        first = newnode;
        last = newnode;
      }
      else
      {
        last->linke = newnode;
        last = newnode;
      }
    }
    else
    {
      newnode->info = array[i];
      newnode->link = first;
      first = newnode;
      if (last==NUL)
        last = first;
    }
  }
}
```

University Of Bahrain IT College, Computer Science Dept. ITCS 215 Data Structure Quiz#4 (Chapter 7)	ID: _ Name Section:

10

Write a function called `removeConsecutiveElem` that accepts a stack ***S*** of type `stackType` and removes repeated consecutive elements from stack **S**. If the stack **S** is empty do nothing.

The function prototype is:

```
void        removeConsecutiveElem( StackType<Type>& S );
```

Example:

Stack **S** before function call : A B B K D D D W F F F W K K M

Stack **S** after function call : A B K D W F W K M

```
void removeConsecutiveElem( stackType<Type> &S)
{   Type first, next;
    StackType ST;
    while(!S.isEmpty())
    {   first = S.Top();
        S.pop();
??      next = S.Top()        → you would get an error here
        if(first != next)
            ST.push(S.Top());
    }
    while(!ST.isEmpty())
    {
        S.push(ST.Top());
        ST.pop();
    }
}
```

University Of Bahrain
IT College, Computer Science Dept.
ITCS 215 Data Structure
Quiz#5 (Chapter 8)

ID :

Name:

Section:

10

What is the output of the following program?

```
#include<iostream>
using namespace std;

void    main( )
{
queueType<int>      qu1(8), qu2(9);
qu1.addQueue( 3 );
int     x = qu1.front( );
qu1.addQueue( 5 );
int     y  = qu1.back( );

        for ( int i=0; i < 3; i++) {
                qu1.addQueue( x + y );
                qu2.addQueue( y – x );
                x = y;
                y = qu1.back( );
        }

while( !qu1.isEmpty( ) ) {
        cout<< qu1.front( ) << " ";
        qu1.deleteQueue( );
}
if ( !qu2.isEmpty() )
        cout<<qu2.front( ) <<" "<<qu2.back( ) <<endl;
}
```

x = 3 8 8 13
y = 8 8 13 21

8 qu1: 3 5 8 13 21
9 qu2: 2 3 5
front
Back

→ 3 5 8 13 21 2 5